```
package main

import (
	"os"
	"github.com/nsf/termbox-go"
)

type
	board struct {
		field [9]rune     // состяние игрового поля
	//каждая клетка содержит X, 0 или точку, точка соответствует пустой клетке
		xc, yc int        // координаты центральной клетки в окне
	}

func InitBoard(x, y int) board {
	return board{field: [9]rune{'.','.','.','.','.','.','.','.','.'}, xc:x, yc:y}
	//  0 1 2
	//  3 4 5
	//  6 7 8
}

func (a board) Result() rune {
	// Возвращает:    X, если выиграли X
	//                0, если выиграли 0
	//                =, если победителя нет, а доска заполнена  - ничья
	//                C, если игра продолжается
	switch {
		case  a.field[0] == a.field[1] && a.field[1] == a.field[2] &&
		      a.field[0] != '.': return a.field[0]
		case  a.field[3]==a.field[4] && a.field[4]==a.field[5] &&
		      a.field[3]!='.': return a.field[3]
		case  a.field[6]==a.field[7] && a.field[7]==a.field[8] &&
		      a.field[6]!='.': return a.field[6]
		case  a.field[0]==a.field[3] && a.field[3]==a.field[6] &&
		      a.field[0]!='.': return a.field[0]
		case  a.field[1]==a.field[4] && a.field[4]==a.field[7] &&
		      a.field[1]!='.': return a.field[1]
		case  a.field[2]==a.field[5] && a.field[5]==a.field[8] &&
		      a.field[2]!='.': return a.field[2]
		case  a.field[0]==a.field[4] && a.field[4]==a.field[8] &&
		      a.field[0]!='.': return a.field[0]
		case  a.field[2]==a.field[4] && a.field[4]==a.field[6] &&
		      a.field[2]!='.': return a.field[2]
	}
	for _, c:= range(a.field) {
		if c == '.' { return 'C'}  // game continues
	}
	return '='  // draw
}

func (a board) Negative() board {
	// Возвращает "негатив" доски - крестики заменяются на нолики и наоборот
	res:= a
	for i, c := range(res.field) {
		if c == 'X' {
			res.field[i] = '0'
		} else
		if c == '0' {
			res.field[i] = 'X'
		}
	}
	return res
}

```

```
65  func (a board) Output() {
66        termbox.SetCell(a.xc-3, a.yc-3, a.field[0],
67                        termbox.ColorDefault, termbox.ColorDefault)
68        termbox.SetCell(a.xc  , a.yc-3, a.field[1],
69                        termbox.ColorDefault, termbox.ColorDefault)
70        termbox.SetCell(a.xc+3, a.yc-3, a.field[2],
71                        termbox.ColorDefault, termbox.ColorDefault)
72        termbox.SetCell(a.xc-3, a.yc  , a.field[3],
73                        termbox.ColorDefault, termbox.ColorDefault)
74        termbox.SetCell(a.xc  , a.yc  , a.field[4],
75                        termbox.ColorDefault, termbox.ColorDefault)
76        termbox.SetCell(a.xc+3, a.yc  , a.field[5],
77                        termbox.ColorDefault, termbox.ColorDefault)
78        termbox.SetCell(a.xc-3, a.yc+3, a.field[6],
79                        termbox.ColorDefault, termbox.ColorDefault)
80        termbox.SetCell(a.xc  , a.yc+3, a.field[7],
81                        termbox.ColorDefault, termbox.ColorDefault)
82        termbox.SetCell(a.xc+3, a.yc+3, a.field[8],
83                        termbox.ColorDefault, termbox.ColorDefault)
84        termbox.Flush()
85  }
86
87  func (a *board) GetMove (sign rune) {  // sign = 'X' or '0'
88        // Ожидает хода игрока - нажатия мышкой на пустую клетку игрового поля
89        // Ставит в выбранную клетку соответствующий знак - крестик или нолик
90        var (
91              exit bool = false
92              pos int
93        )
94        for !exit {
95              ev := termbox.PollEvent()
96              if ev.Type == termbox.EventMouse && ev.Key == termbox.MouseRelease {
97                    exit = true
98                    dx, dy:= ev.MouseX - (*a).xc, ev.MouseY - (*a).yc
99                    switch {
100                         case dx>=-4 && dx<=-2 && dy>=-4 && dy<=-2: pos = 0
101                         case dx>=-1 && dx<= 1 && dy>=-4 && dy<=-2: pos = 1
102                         case dx>= 2 && dx<= 4 && dy>=-4 && dy<=-2: pos = 2
103                         case dx>=-4 && dx<=-2 && dy>=-1 && dy<= 1: pos = 3
104                         case dx>=-1 && dx<= 1 && dy>=-1 && dy<= 1: pos = 4
105                         case dx>= 2 && dx<= 4 && dy>=-1 && dy<= 1: pos = 5
106                         case dx>=-4 && dx<=-2 && dy>= 2 && dy<= 4: pos = 6
107                         case dx>=-1 && dx<= 1 && dy>= 2 && dy<= 4: pos = 7
108                         case dx>= 2 && dx<= 4 && dy>= 2 && dy<= 4: pos = 8
109                         default: exit = false
110                   }
111             }
112             if exit && (*a).field[pos] != '.' {
113                   exit = false
114             }
115       }
116       (*a).field[pos] = sign
117 }
118
```

```
119  func (a board) Estimate() (value rune, bestMove int) {
120        // Оценочная функция
121        value = a.Result()
122        if value=='X' || value=='0' || value=='=' { return value, -1 }
123        value = '0'
124        bestMove = 0
125        for i, c:= range(a.field) {
126              if c == '.' {
127                    a.field[i] = 'X'
128                    e, _:= a.Negative().Estimate()
129                    a.field[i] = '.'
130                    switch e {
131                    case '=':
132                          value = '='
133                          bestMove = i
134                    case '0':
135                          return 'X', i
136                    }
137              }
138        }
139        return value, bestMove
140  }
141
142  func GetFirst() (First, Second string) {
143        termbox.Clear(termbox.ColorDefault, termbox.ColorDefault)
144        w, h := termbox.Size()
145        OutputText("Who starts the game?", w/2 - 10, h/2 - 2)
146        OutputText("Human", w/2 - 10, h/2 + 2)
147        OutputText("Computer", w/2 + 5, h/2 + 2)
148        for {
149              ev := termbox.PollEvent()
150              if ev.Type == termbox.EventMouse && ev.Key == termbox.MouseLeft {
151                    if ev.MouseY >= h/2 {
152                          if ev.MouseX > w/2 {
153                                termbox.Clear( termbox.ColorDefault,
154                                               termbox.ColorDefault)
155                                return "Computer", "Human"
156                          }
157                          if ev.MouseX < w/2 {
158                                termbox.Clear(termbox.ColorDefault,
159                                              termbox.ColorDefault)
160                                return "Human", "Computer"
161                          }
162                    }
163              }
164        }
165  }
166
167  func OutputText(Message string, x0, y0 int) {
168        x, y:= x0, y0
169        for _, c:= range([]rune(Message)) {
170              termbox.SetCell(x, y, c, termbox.ColorDefault, termbox.ColorDefault)
171              x++
172        }
173        termbox.Flush()
174  }
175
```

```
func main() {
	err := termbox.Init()
	if err != nil { // Ошибка инициализации termbox
		os.Exit(1)
	}
	defer termbox.Close()
	termbox.HideCursor()
	termbox.SetInputMode(termbox.InputEsc + termbox.InputMouse)

	First, Second := GetFirst()
	currentSign:= 'X'
	activePlayer:= First

	w, h := termbox.Size()
	f:= InitBoard(w/2, h/2) // состояние игры: положение на
                                    // поле и координаты центра доски
	f.Output()

	for f.Result()=='C' {
		if activePlayer == "Human" {
			f.GetMove(currentSign)
			f.Output()
		} else {
		// activePlayer    == "Computer"
			var bestMove int
			if currentSign == 'X' {
				_, bestMove = f.Estimate()
			} else {
				_, bestMove = f.Negative().Estimate()
			}
			f.field[bestMove] = currentSign
		}
		f.Output()
		if currentSign == 'X' {
			currentSign = '0'
		} else {
			currentSign = 'X'
		}
		if activePlayer == "Human" {
			activePlayer = "Computer"
		} else {
			activePlayer = "Human"
		}
	}
	var WinMessage string
	switch f.Result() {
	case 'X':
		WinMessage = First + " won"
	case '0':
		WinMessage = Second + " won"
	case '=':
		WinMessage = "Draw"
	}
	OutputText(WinMessage, f.xc - len([]rune(WinMessage))/2, f.yc + 8)
	termbox.PollEvent()
}
```